INCREMENTAL AST MAINTENANCE USING WORK AREAS

CROSS REFERENCE TO RELATED APPLICATIONS

[0001] The present application is related to pending U.S. Patent Application No. 09/453,892, filed December 2, 1999 to Roberta Cochrane et al., entitled "Incremental Maintenance of Summary Tables with Complex Grouping Expressions," having (IBM) Docket No. AM9-99-0231; U.S. Patent Application No. 10/335,376 filed December 30, 2002 to Roberta Cochrane et al., entitled "System and Method For Incrementally Maintaining Non-Distributive Aggregate Functions In a Relational Database," having (IBM) Docket No. ARC920030030US1; and Canada Patent Application No. 241 4983 filed in Canada on December 23, 2003, entitled "Independent Deferred Incremental Refresh of Materialized Views", having (IBM) Docket No. CA9-2002-0036. The foregoing applications are assigned to the present assignee, and are all incorporated herein by reference.

BACKGROUND OF THE INVENTION

Field of the Invention

[0002] The present invention generally relates to incrementally maintaining algebraic functions in automatic summary tables (ASTs) of at least one relational database.

Description of the Related Art

[0003] Within this application several publications are referenced by Arabic numerals within parentheses. Full citations for these, and other, publications may be found at the end of the specification immediately preceding the claims. The disclosures of all these publications in their entireties are hereby expressly incorporated by reference into the present application for the purposes of indicating the background of the present invention and illustrating the state of the art.

[0004] Automatic Summary Tables (ASTs), which are also known as materialized views physically, store derived data from several relational database tables. ASTs are of important usage to various applications to improve query performance [3] such as OLAP or data mining applications. However, as base tables change over time, one of the important issues of ASTs is to maintain the ASTs extent upon base changes. Since re-computation from base tables is usually expensive, incremental maintenance of ASTs is often a better solution in terms of performance [1, 6, 7].

[0005] Materialized views, or Automatic Summary Tables (ASTs), are increasingly being used to facilitate the analysis of the large amounts of data being collected in relational databases. The use of ASTs can significantly reduce the execution time of a query. This reduction in execution time is particularly significant for databases with sizes in the terabyte to petabyte range. Such queries tend to be extremely complex and can involve a large number of join and grouping operations.

[0006] One major advantage of using ASTs is that they are precomputed once and subsequently can be used multiple times to quickly answer complex queries. When base is relations are modified, these modifications must be propagated to the affected ASTs. Unfortunately, using current techniques, the systems can only incrementally update a restricted set of ASTs, e.g., those only containing distributive aggregate functions. The remainder must be fully recomputed. Previous work has studied the problem of incremental view maintenance in which all the necessary changes for the AST are computed based only on the modifications to the base table (and the corresponding values in the AST). This process is called incremental view maintenance and many commercial products support it.

[0007] Due to the complexity of the queries and the magnitude of the data, recomputation of ASTs in large-scale databases is prohibitive. Since the set of updates to the base tables is usually only some small percentage of those tables, incremental maintenance of an AST is usually much quicker than full recomputation. For example, a typical warehouse may contain up to six (6) years of data. Daily inserts into a fact table in this warehouse may constitute only about five hundredths of a percent (0.05%) of the entire size of the table. When updates occur in the base data, the system determines which ASTs are affected and propagates the changes through the AST definitions to produce the delta changes. It then applies these deltas to their respective ASTs. If an AST is automatically refreshed in the same unit of work as the changes to the underlying base data are applied, then the maintenance is considered immediate. Otherwise, it is deferred.

SUMMARY OF THE INVENTION

[0008] The present invention will be better appreciated and understood when considered in conjunction with the following description and the accompanying drawings. It should be understood, however, that the following description, while indicating preferred embodiments of the present invention and numerous specific details thereof, is given by way of illustration and not of limitation. Many changes and modifications may be made within the scope of the present invention without departing from the spirit thereof, and the invention includes all such modifications.

[0009] The invention provides a method and system for incrementally maintaining algebraic functions in automatic summary tables (ASTs) of at least one relational database. More specifically, the invention associates a work area with each algebraic function in each AST. Each work area is comprised of one or more variables. The invention populates the variables within each work area for each algebraic function when each AST is created and when each AST is updated. The invention maintains each work area by adding and subtracting to and from its associated variables when the associated data changes in the relational database. The functions that compute the variables of the work areas are distributive and thus incrementally maintainable. The invention computes and recomputes the algebraic function within an AST after the updates have changed one or more of the variables of its workarea.

[0010] In one embodiment, multiple algebraic functions may share the same work area. For example, multiple algebraic functions may share the same work areas when the work area requirements of the algebraic functions match exactly, match partially, or intersect.

[0011] Thus, with the invention, the association of a work area for each algebraic function makes the aggregate algebraic function incrementally maintainable. Also, the invention designs a framework in the query compilation phase for incremental maintenance of the algebraic functions in an AST. Also, the invention generalizes the solutions and framework to any algebraic functions, even user defined aggregates.

BRIEF DESCRIPTION OF THE DRAWINGS

[0012] The invention will be better understood from the following detailed description with reference to the drawings, in which:

[0013] Figure 1 is a schematic diagram of a system architecture;

[0014] Figure 2 is a schematic diagram of a framework for incremental maintenance of algebraic functions in AST; and

[0015] Figure 3 is a flow diagram illustrating a preferred method of the invention.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS OF THE INVENTION

[0016] The present invention and the various features and advantageous details thereof are explained more fully with reference to the nonlimiting embodiments that are illustrated in the accompanying drawings and detailed in the following description. It should be noted that the features illustrated in the drawings are not necessarily drawn to scale. Descriptions of well-known components and processing techniques are omitted so as to not unnecessarily obscure the present invention in detail.

[0017] ASTs usually contain aggregates or super aggregates. A super-aggregate is a SQL language clause that supports the computation of measures for different levels of hierarchy. There are three types of aggregate functions, namely, distributive, algebraic and holistic. Distributive functions include, for example, SUM and COUNT. The aggregate functions of this category are characterized as being incrementally maintainable. For example, given two SUMs of two datasets, the invention can compute the new SUM simply by adding these two values.

[0018] The second category of the aggregate functions is algebraic functions, such as standard deviation, variance, covariance, correlation, and regression family functions. Such aggregate functions are not directly incrementally maintainable, e.g., given two standard deviations, it is not possible to compute the new standard deviation simply based on these two values. However, by using some additional bounded space, these functions can be made incrementally maintainable.

[0019] How to maintain such aggregates introduces additional complexity [4, 6]. This application provides incremental maintenance of one type of aggregate functions, called algebraic functions, such as standard deviation, regression functions, etc. However, such algebraic functions are not directly incrementally maintainable by themselves. Instead, the invention associates some additional bounded storage space called a "work area", with each algebraic function. Now the problem of the incremental maintenance of the algebraic functions becomes the incremental maintenance of its corresponding work area. By properly defining the work area maintenance functions, the invention is able to incrementally maintain the algebraic functions.

[0020] The present invention associates a work area with algebraic functions for incremental maintenance. Shown below is a brief illustration of the techniques of the invention. Consider an AST with an average aggregate function Average(X) as shown:

```
SELECT C.c1, avg(C.c2), count(*) FROM C GROUP BY C.c1;
```

Notice that avg(C.c2) is not incrementally maintainable. Therefore, the invention adds an additional column, work area (SUM(C.c2), COUNT(C.c2)) associated with the avg(C.c2) column. Since SUM and COUNT are incrementally maintainable, computing the new average simply by dividing the maintained work area SUM by work area COUNT is the preferred direction.

[0021] Thus, with the invention, the association of a work area for each algebraic function makes the aggregate algebraic function incrementally maintainable. Also, the invention designs a framework in the query compilation phase for incremental maintenance of the algebraic functions in AST. Also, the invention generalizes the solutions and framework to any algebraic functions, even user defined aggregates.

[0022] Specific embodiments of the invention will now be further described by the following nonlimiting examples, which will serve to illustrate in some detail, various features of significance. The examples are intended merely to facilitate an understanding of ways in which the invention may be practiced and to further enable those of skill in the art to practice the invention. Accordingly, the examples should not be construed as limiting the scope of the invention.

[0023] Referring now to the drawings, there are shown preferred embodiments of the method and structures according to the present invention. Referring initially to FIG. 1, the system architecture is shown and is generally designated 10. FIG. 1 shows that the system 10 includes one or more processors 12, 14, 16 that are connected to one or more data storage devices 18, 20, 22, such as disk drives, in which one or more relational databases are stored. In a preferred embodiment, each processor 12, 14, 16 includes a maintenance module 24, 26, 28 for incrementally maintaining materialized views of the relational databases stored in the storage devices 18,20,22.

[0024] Preferably, each of the processors 12, 14, 16 utilize a standard operator interface, e.g., IMS/DB/DC, CICS, TSO, OS/2 or other similar interface, to transmit electrical signals to and from the processors 12, 14, 16. The electrical signals represent commands for performing various search and retrieval functions, i.e., termed queries, against the databases stored in the data storage devices 18, 20, 22. Preferably, these queries conform to the Structured Query Language (SQL) standard, and invoke functions performed by Relational Database Management

System (RDBMS) software. In a preferred embodiment, the RDBMS software comprises the DB2 product offered by IBM for the MVS, OS/2, UNIX, or WINDOWS NT operating systems. Those skilled in the art will recognize, however, that the present invention has application to any RDBMS software.

[0025] It is to be understood that, in a preferred embodiment, each processor 12, 14, 16 includes a series of computer-executable instructions, as described below, which will allow each processor to provide incremental maintenance for materialized views in the relational databases residing on the data storage devices 18, 20, 22. These instructions may reside, for example, in the maintenance modules 24, 26, 28 of the processors 12, 14, 16, which can simply be a portion of the random access memory (RAM) of the processors 12, 14, 16.

[0026] Alternatively, the instructions may be contained on a data storage device with a computer readable medium, such as a computer diskette. Or, the instructions may be stored on a magnetic tape, hard disk drive, electronic read-only memory (ROM), optical storage device, or other appropriate data storage device or transmitting device thereby making a computer program product, i.e., an article of manufacture according to the invention. In an illustrative embodiment of the invention, the computer-executable instructions may be lines of C++ compatible code.

[0027] The flow charts herein illustrate the structure of the logic of the present invention as embodied in computer program software. Those skilled in the art will appreciate that the flow charts illustrate the structures of computer program code elements including logic circuits on an integrated circuit that function according to this invention. Manifestly, the invention is practiced in its essential embodiment by a machine component that renders the program elements in a form that instructs a digital processing apparatus (that is, a computer) to perform a sequence of function steps corresponding to those shown.

[0028] Figure 2 presents the referenced framework for incremental maintenance of algebraic functions in AST. As shown in Figure 2, basically, AST maintenance is broken into two compilation phases, namely, propagation phase 114 and apply phase 110. At execution time the results of the propagation phase 114 of compilation computes the combined delta effects 112 of base updates 116, 118. The execution time results of the apply phase 100 of compilation integrates such delta effects 112 into the AST 120.

[0029] The shaded modules 100, 102, 104 in Figure 2 show the extension of the framework for maintaining algebraic function. First, during the propagation phase 114, the invention creates a corresponding work area 102 as a column for each algebraic function. Second, these work areas will be added and populated when the AST 120 is created 104 and when the AST is refreshed 104. Third, during the apply phase 110, the work area 120 is maintained by the maintenance functions specific to the work area 100. The algebraic function is then in turn computed from the work area 100.

[0030] In addition, Figure 3 is a flowchart illustrating the processing achieved with the invention. More specifically, the invention associates a work area with each algebraic function in each AST 30 and then populates variables of each algebraic function within each work area 32 when each AST is created and when each AST is updated. The invention maintains each work area 34 by adding and subtracting to and from its associated variables when the associated data changes in the relational database. The functions that compute the variables of the work areas are distributive and thus incrementally maintainable. The invention computes and recomputes the algebraic function 36 within an AST after the updates have changed one or more of the variables of its workarea.

[0031] As previously mentioned, to maintain the algebraic function, the invention first maintains its corresponding work area and then computes the algebraic function from the new work area. Following is the description of those maintenance functions in detail and their properties.

[0032] To maintain the work area, three basic maintenance functions are used: recompute, plus and minus. In one example, these functions are defined is as follows. For recompute: WA=Recomp(S). Therefore, the invention recomputes the work area (WA) given a dataset S. Plus is given two datasets S_1 and S_2 and their corresponding work areas WA_{S1} and WA_{S2}, i.e., $WA_{S1 \cup S2}$= Plus(WA_{S1}, WA_{S2}). Minus is given two datasets S_1 and S_2, $S_1 \supseteq S_2$, and their corresponding work areas WA_{S1} and WA_{S2}. For example, the invention can avoid recomputing the work area $WA_{S1\text{-}S2}$ by introducing a new function Minus on WA_{S1} and WA_{S2}, i.e., $WA_{S1\text{-}S2}$= *Minus*(WA_{S1}, WA_{S2}).

For example, average's work area is (SUM, COUNT):

Plus((SUM_1, $COUNT_1$), (SUM_2, $COUNT_2$)) = ($SUM_1 + SUM_2$, $COUNT_1 + COUNT_2$) and *Minus*((SUM_1, $COUNT_1$), (SUM_2, $COUNT_2$)) = ($SUM_1 - SUM_2$, $COUNT_1 - COUNT_2$).

[0033] Plus is used for inserts while minus is used for deletes. Note that the plus function can be used for the parallel computation of a work area to combine the work areas from different data partitions. Given an update to the database that involves both inserts and deletes to the base data, the new value of the work area in the AST WA^{AST} is computed from the work area of the inserts WA^{Insert} and the work area of the deletes WA^{Delete} by the following formula:

$Minus((Plus(\mathrm{WA}^{\mathrm{AST}}, \mathrm{WA}^{\mathrm{Insert}}), \mathrm{WA}^{\mathrm{Delete}})$

Finally, the algebraic function is computed from the work area that has been maintained. For example, AVG is computed from its work area (SUM, COUNT) by SUM/COUNT.

[0034] Although theoretically incremental maintenance of algebraic functions using the Recomp, Plus, and Minus functions should provide the same results as full recomputation but due to inherent floating point number computation errors, different results may occur. First, the maintained algebraic results may be different from the results after recomputation and these differences may compound over time as multiple changes are made to the base tables and these changes are propagated to the AST. This may not be desirable for some applications. Second, erroneous results may occur around the boundaries. For example, the variance may result in a very small negative number instead of zero due to some floating point computation errors. This is obviously unexpected. .

[0035] To address differences caused by floating point number computations, the invention estimates the error associated with the incremental maintenance procedure. For each variable V of the work area that is required to maintain the algebraic function, another variable VE is added to the work area that contains an estimate of the error in V. The variable VE estimates the error between the value of V and the full recomputation value of V. The functions that compute the error estimate variables VE are distributive and thus also incrementally maintainable. The error estimate variables are maintained incrementally as part of the work area. The estimated error in the algebraic function (versus the full recomputation value) is computed from the error estimate variables VE of the work area. When, for a given group (i.e., row) of the AST, the ratio of (1) the estimated error of the algebraic function to (2) the incrementally computed value of the algebraic function exceeds a threshold (for example, 0.1%), the invention applies the method of selective recomputation [5] to recompute the value of that group from the base table data. The method of selective recomputation is also applied to those groups that violate a boundary condition (e.g. a negative value for variance).

[0036] Notice that some algebraic functions use the same functions and compute the same values for their work areas. For example, regression family functions share the same work area (SUM(X), COUNT(X), SUM(Y), COUNT(Y), VAR(X), VAR(Y), COV(X,Y)). Hence, it is possible to share the work area among algebraic functions. The benefit of such sharing is reducing the space required to create the work areas and saving time to maintain the work areas.

[0037] Basically, there are different types of possible shares. First, an each exact match has two or more work area functions that are exactly the same. Second, a partial match has one work area function that is a subset of another. Third, an intersection has some overlapping between two work area functions, but they are not subset of each other. Such functions can share these work areas because they can be computed from the same work area.

[0038] To implement the invention in a relational database program, such as DB2, for each algebraic function, a work area column (of type Bit Data) which may contain multiple logical fields is created. Thus, the invention provides a work area approach for incremental algebraic AST maintenance, in particular, the association of a work area with each algebraic function. Then, the maintenance of the algebraic function turns to the maintenance of its corresponding work area.

[0039] With the invention, the association of a work area for each algebraic function makes the aggregate algebraic function incrementally maintainable. Also, the invention designs a framework in the query compilation phase for incremental maintenance of the algebraic functions in AST. The invention addresses precision errors of incremental maintenance by incrementally maintaining estimates of the errors. Also, the invention generalizes the solutions and framework to any algebraic functions, even user defined aggregates.

[0040] One benefit of the invention is performance. The invention permits incremental maintenance of ASTs with algebraic functions. Incremental maintenance is often much more efficient than full recomputation. ASTs are used to improve query performance.

[0041] While the invention has been described in terms of preferred embodiments, those skilled in the art will recognize that the invention can be practiced with modification within the spirit and scope of the appended claims.

REFERENCES

1. J. A. Blakeley et al., "Efficiently Updating Materialized Views", *Proceedings of SIGMOD*, pp. 61-71, 1986.

2. J. Gray et al., "Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Total," In *Proceedings of IEEE International Conference on Data Engineering,* pp. 152-159, 1996.

3. A. Gupta et al, "Maintenance of Materialized Views: Problems, Techniques, and Applications," *IEEE Data Engineering Bulletin, Special Issue on Materialized Views and Warehousing*, 18(2): 3-19, 1995.

4. I. Mumick et al., "Maintenance of Data Cubes and Summary Tables in a Warehouse," In *Proceedings of SIGMOD*, May, 1997.

5. T. Palpanas et al., "Selective Recomputation for Non-Distributive Aggregate Functions," In *Proceedings of VLDB*, 2002.

6. K. Salem et al., 'How to Roll a Join: Asynchronous Incremental View Maintenance," In *Proceedings of SIGMOD*, PP. 129-140, 2000.

7. Y. Zhuge et al., "View Maintenance in a Warehousing Environment," In *Proceedings of SIGMOD*, PP. 316-327, May, 1995.